1

CPU Profile

CONTENTS

Figures

Tables

167

168   Foreword

169   The *CPU Profile* (DSP1022) was prepared by the Physical Platform Profiles Working Group of the DMTF.

170   DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems
171   management and interoperability. For information about the DMTF, see http://www.dmtf.org.

Acknowledgments

173   The DMTF acknowledges the following individuals for their contributions to this document:

174   Editors:

175   • Jon Hass – Dell

176   • Jeff Hilland – Hewlett-Packard Company

177   • John Leung - Intel

178   • Khachatur Papanyan – Dell

179   Contributors:

180   • Jeff Lynch – IBM

181   • Aaron Merkin – IBM

182   • Christina Shaw – Hewlett-Packard Company

183   • Perry Vincent – Intel

184   Introduction

185   The information in this specification should be sufficient for a provider or consumer of this data to identify
186   unambiguously the classes, properties, methods, and values that shall be instantiated and manipulated to
187   represent and manage the processor components of systems and subsystems modeled using the DMTF
188   Common Information Model (CIM) core and extended model definitions.

189   The target audience for this specification is implementers who are writing CIM-based providers or
190   consumers of management interfaces that represent the component described in this document.

191

192 # CPU Profile

1   Scope

194 The *CPU Profile* extends the management capability of referencing profiles by adding the capability to
195 represent CPUs or processors in a managed system. CPU cache memory and associations with CPU
196 physical aspects, as well as profile implementation version information, are modeled in this profile.

2   Normative references

198 The following referenced documents are indispensable for the application of this document. For dated or
199 versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies.
200 For references without a date or version, the latest published edition of the referenced document
201 (including any corrigenda or DMTF update versions) applies.

202 DMTF DSP0004, *CIM Infrastructure Specification 2.5,*
203 http://www.dmtf.org/standards/published_documents/DSP0004_2.5.pdf

204 DMTF DSP0134, *System Management BIOS (SMBIOS) Reference Specification 2.6,*
205 http://www.dmtf.org/standards/published_documents/DSP0134_2.6.pdf

206 DMTF DSP0200, *CIM Operations over HTTP 1.3,*
207 http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf

208 DMTF DSP1001, *Management Profile Specification Usage Guide 1.0,*
209 http://www.dmtf.org/standards/published_documents/DSP1001_1.0.pdf

210 DMTF DSP1011, *Physical Asset Profile 1.0*,
211 http://www.dmtf.org/standards/published_documents/DSP1011_1.0.pdf

212 DMTF DSP1033, *Profile Registration Profile 1.0,*
213 http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf

214 IETF RFC5234, *Augmented BNF for Syntax Specifications: ABNF,* January 2008,
215 http://www.rfc-editor.org/rfc/rfc5234.txt

216 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*

3   Terms and definitions

218 In this document, some terms have a specific meaning beyond the normal English meaning. Those terms
219 are defined in this clause.

220 The terms "shall" ("required"), "shall not," "should" ("recommended"), "should not" ("not recommended"),
221 "may," "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described
222 in ISO/IEC Directives, Part 2, Annex H. The terms in parenthesis are alternatives for the preceding term,
223 for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that
224 ISO/IEC Directives, Part 2, Annex H specifies additional alternatives. Occurrences of such additional
225 alternatives shall be interpreted in their normal English meaning.

226 The terms "clause," "subclause," "paragraph," and "annex" in this document are to be interpreted as
227 described in ISO/IEC Directives, Part 2, Clause 5.

228 The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC
229 Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do
230 not contain normative content. Notes and examples are always informative elements.

231 For the purposes of this document, the following terms and definitions apply. The terms defined in
232 DSP0004, DSP0200, DSP1001, and DSP1033 also apply to this document.

233 **3.1**
234 **Cache Memory**
235 indicates the instance of CIM_Memory that represents the cache memory for the processor

236 **3.2**
237 **Host Processor**
238 indicates the instance of CIM_Processor that represents the processor that hosts the processor core

239 **3.3**
240 **Threading Processor Core**
241 indicates the instance of CIM_ProcessorCore that represents the processor core that enables the
242 hardware threading

243 4   Symbols and abbreviated terms

244 **4.1**
245 **CPU**
246 central processing unit

247 5   Synopsis

248 **Profile Name:** *CPU*

249 **Version:** 1.0.2

250 **Organization:** DMTF

251 **CIM Schema Version:** 2.19

252 **Central Class:** CIM_Processor

253 **Scoping Class:** CIM_ComputerSystem

254 The *CPU Profile* is a component profile that extends the management capability of referencing profiles by
255 adding the capability to represent CPUs or processors in a managed system.

256 **Table 1 – Related Profiles**

Profile Name	Organization	Version	Requirement	Description
Physical Asset	DMTF	1.0	Optional	See 7.9.
Profile Registration	DMTF	1.0	Mandatory	

257 # 6 Description

258 The *CPU Profile* describes CPU or processor devices and associated cache memory used in managed
259 systems.

260 The profile could manage the following capabilities of a typical computer system:

261 • A computer system can have one or more processors, which may be individually enabled or
262 disabled.

263 • A processor can contain one or more processor cores, which may be individually enabled or
264 disabled.

265 • A processor core can contain one or more hardware threads, which may be individually enabled or
266 disabled

267 Figure 1 represents the class schema for the *CPU Profile*. For simplicity, the prefix CIM_ has been
268 removed from the names of the classes.

269 The CIM_Processor class represents a group of processor cores; the CIM_ProcessorCapabilities class
270 describes the capabilities of the processor. The CIM_Processor may be associated to one or more of
271 instances of CIM_ProcessorCore, through the CIM_ConcreteComponent association.

272 The CIM_ProcessorCore class represents a processing execution unit. The CIM_ProcessorCore may be
273 associated to one or more instances of CIM_HardwareThread, through the CIM_ConcreteComponent
274 association.

275 The CIM_HardwareThread class represents a hardware thread, a mechanism by which a processing
276 execute unit is made to appear as multiple processing units (each called a virtual core).

277 The CIM_Memory class represents cache memory. CIM_Memory may be associated to either
278 CIM_Processor or CIM_ProcessorCore, through the CIM_AssociatedCacheMemory association.

279 The CIM_Chip class represents the physical aspects of a processor. The CIM_PhysicalMemory
280 represents the cache memory, when the cache memory is off-chip/external.
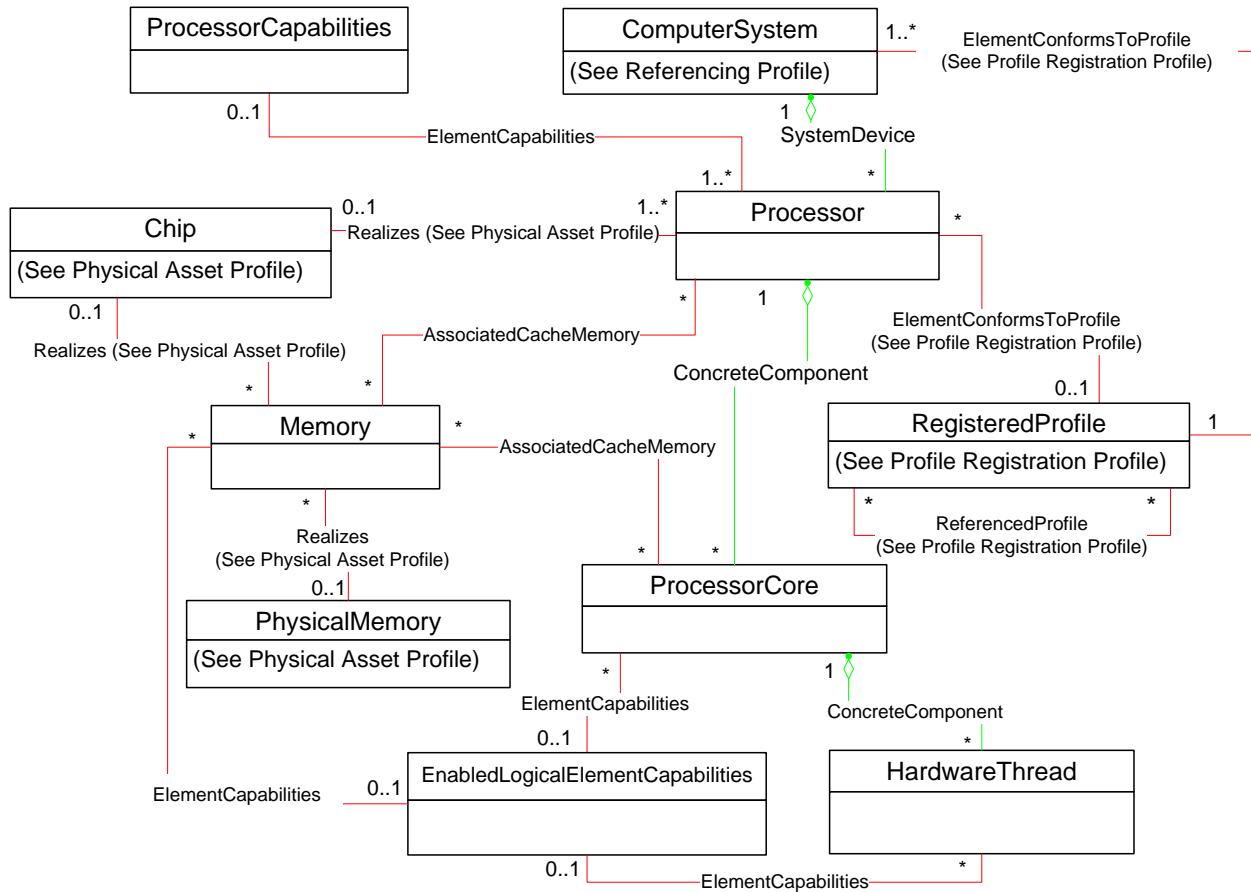
Figure 1 – CPU Profile: Class Diagram

7   Implementation

This clause details the requirements related to the arrangement of instances and their properties for implementations of this profile. Methods are listed in clause 8 ("Methods"), and properties are listed in clause 10 ("CIM Elements").

7.1   CIM_Processor

Zero or more instances of CIM_Processor shall be instantiated.

7.2   Processor capabilities

The CIM_ProcessorCapabilities class may be instantiated to represent the processor capabilities. Only one instance of CIM_ProcessorCapabilities shall be associated with a given instance of CIM_Processor through an instance of CIM_ElementCapabilities.

7.2.1   Multi-Core or Multi-Thread processor capabilities

When modeling the capabilities of a multi-core or multi-thread processor, the CIM_ProcessorCapabilities class shall be instantiated and associated to the instance of CIM_Processor that represents the multi-core or multi-thread processor.

297 The properties of CIM_ProcessorCapabilities described in the "CIM_ProcessorCapabilities Properties"
298 column in Table 2 are defined in terms of the [DSP0134](#) Processor Information structure to provide
299 meaningful context for the interpretation of the properties. The mappings specified in Table 2 shall be
300 used. The underlying represented system does not need to support [DSP0134](#).

301 **Table 2 – CIM_ProcessorCapabilities Properties mapping to SMBIOS equivalence**

CIM_ProcessorCapabilities Properties	SMBIOS Structure Name	SMBIOS Structure Description
NumberOfProcessorCores	Core Count	Number of cores per processor socket
NumberOfHardwareThreads	Thread Count	Number of threads per processor socket

302 ### 7.2.2 Single-Core and Single-Thread processor capabilities

303 When modeling the capabilities of a single-core and single-thread processor, the
304 CIM_ProcessorCapabilities may not be instantiated.

305 When no instance of CIM_ProcessorCapabilities is associated with the instance of CIM_Processor that
306 represents the processor, the processor is a single-core and single-thread processor.

307 When an instance of CIM_ProcessorCapabilities is associated with the instance of CIM_Processor that
308 represents the single-core and single-thread processor, the following requirements apply:

309 • The CIM_ProcessorCapabilities.NumberOfProcessorCores property shall have a value of 1.

310 • The CIM_ProcessorCapabilities.NumberOfHardwareThreads property shall have a value of 1.

311 ### 7.2.3 CIM_ProcessorCapabilities.RequestedStatesSupported

312 The RequestedStatesSupported property is an array that contains the supported requested states for the
313 instance of CIM_Processor. This property shall be the super set of the values to be used as the
314 RequestedState parameter in the RequestStateChange( ) method (see 8.1). The value of the
315 CIM_ProcessorCapabilities.RequestedStatesSupported property shall be an empty array or contain any
316 combination of the following values: 2 (Enabled), 3 (Disabled), or 11 (Reset).

317 ### 7.2.4 CIM_ProcessorCapabilities.ElementNameEditSupported

318 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports
319 client modification of the CIM_Processor.ElementName property.

320 ### 7.2.5 CIM_ProcessorCapabilities.MaxElementNameLen

321 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported
322 property has a value of TRUE.

323 ## 7.3 Processor state management

324 Processor state management requires that the CIM_Processor.RequestStateChange( ) method be
325 supported (see 8.1) and the value of the CIM_Processor.RequestedState property not match 12 (Not
326 Applicable).

327 ### 7.3.1 Processor state management support

328 When the instance of CIM_ProcessorCapabilities does not exist, processor state management shall not
329 be supported.

330   When the value of the CIM_ProcessorCapabilities.RequestedStatesSupported property of the associated
331   CIM_ProcessorCapabilities instance is an empty array, processor state management shall not be
332   supported.

333   When the value of the CIM_ProcessorCapabilities.RequestedStatesSupported property of the associated
334   CIM_ProcessorCapabilities instance is not an empty array, processor state management shall be
335   supported.

336   7.4   CIM_Processor.RequestedState

337   The CIM_Processor.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No Change),
338   or a value contained in the CIM_ProcessorCapabilities.RequestedStatesSupported property array of the
339   associated CIM_ProcessorCapabilities instance (see 7.2.2).

340   When processor state management is supported and the RequestStateChange( ) method is successfully
341   executed, the RequestedState property shall be set to the value of the RequestedState parameter of the
342   RequestStateChange( ) method. After the RequestStateChange( ) method has successfully executed, the
343   RequestedState and EnabledState properties shall have equal values with the exception of the
344   transitional requested state 11 (Reset). The value of the RequestedState property may also change as a
345   result of a request for a change to the processor's enabled state by a non-CIM implementation.

346   7.4.1   RequestedState — 12 (Not Applicable) value

347   When processor state management is not supported, the value of the CIM_Processor.RequestedState
348   property shall be 12 (Not Applicable).

349   7.4.2   RequestedState — 5 (No Change) value

350   When processor state management is supported, the initial value of the CIM_Processor.RequestedState
351   property shall be 5 (No Change).

352   7.5   Modeling the current enabled state of the processor

353   The current enabled state of the processor is described by the CIM_Processor.CPUStatus and
354   CIM_Processor.EnabledState properties. Clauses 7.5.1 and 7.5.2 detail the requirements for
355   implementing these two properties.

356 **7.5.1 CIM_Processor.CPUStatus**

357 Table 3 describes the mapping between the values of the CIM_Processor.CPUStatus property and the
358 corresponding description of the state of the processor. The CPUStatus property shall match the values
359 that are specified in Table 3. When the RequestStateChange( ) method executes but does not complete
360 successfully, or the processor is in an indeterminate state, the CPUStatus property shall have value of 0
361 (Unknown). The value of this property may also change as a result of a change to the processor's
362 enabled state by a non-CIM implementation.

363 **Table 3 – CIM_Processor.CPUStatus Value Descriptions**

Value	Description	Extended Description
0	Unknown	Processor state is indeterminate, or the processor state management is not supported.
1	CPU Enabled	Processor shall be enabled.
2	CPU Disabled by User	Processor shall be disabled through client configuration, which may occur through client invocation of the RequestStateChange( ) method or through a non-CIM implementation such as BIOS.
3	CPU Disabled By BIOS (POST Error)	Processor shall be disabled due to a POST error.
4	CPU Is Idle, waiting to be enabled	Processor shall be enabled but idling.

364 **7.5.2 CIM_Processor.EnabledState**

365 The CIM_Processor.EnabledState property shall be implemented in addition to the
366 CIM_Processor.CPUStatus property. When the CPUStatus property has a value that matches a value in
367 the "CPUStatus Value" column in Table 4, the EnabledState property shall have a value that matches a
368 value in the "EnabledState Value" column in the same row in the table.

369 **Table 4 – Mapping for CPUStatus Property and EnabledState Property Values**

CPUStatus Value	Description	EnabledState Value	Description
0	Unknown	0 or 5	Unknown or Not Applicable
1	CPU Enabled	2	Enabled
2	CPU Disabled by User	3	Disabled
3	CPU Disabled By BIOS (POST Error)	3	Disabled
4	CPU Is Idle, waiting to be enabled	2	Enabled

370 **7.6 Modeling individual processor cores**

371 Modeling the individual processor cores is optional functionality. When individual processor cores are
372 modeled, the implementation shall instantiate an instance of CIM_ProcessorCore to represent each
373 processor core. All the requirements in this clause and its subclauses are applicable when an
374 implementation instantiates the CIM_ProcessorCore class.

375 Each instance of CIM_ProcessorCore shall be associated through an instance of
376 CIM_ConcreteComponent to only one instance of CIM_Processor that represents the processor (Host
377 Processor) that hosts the processor core.

378 The number of instances of CIM_ProcessorCore associated with the Host Processor shall be equal to the
379 value of the CIM_ProcessorCapabilities.NumberOfProcessorCores property of the instance of
380 CIM_ProcessorCapabilities that is associated with the Host Processor.

7.6.1 Processor core capabilities

382 The CIM_EnabledLogicalElementCapabilities class may be used to model the capabilities of processor
383 cores. When the CIM_EnabledLogicalElementCapabilities class is instantiated to represent the processor
384 core capabilities, the instance of CIM_EnabledLogicalElementCapabilities shall be associated with the
385 CIM_ProcessorCore instance through an instance of CIM_ElementCapabilities and used for advertising
386 the capabilities of the CIM_ProcessorCore instance.

387 There shall be at most one instance of CIM_EnabledLogicalElementCapabilities associated with a given
388 instance of CIM_ProcessorCore.

7.6.1.1 CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported

390 The RequestedStatesSupported property is an array that contains the supported requested states for the
391 instance of CIM_ProcessorCore. This property shall be the super set of the values to be used as the
392 RequestedState parameter in the RequestStateChange( ) method (see 8.2). The value of the
393 RequestedStatesSupported property shall be an empty array or contain any combination of the following
394 values: 2 (Enabled), 3 (Disabled), or 11 (Reset).

7.6.1.2 CIM_EnabledLogicalElementCapabilities.ElementNameEditSupported

396 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports
397 client modification of the CIM_ProcessorCore.ElementName property.

7.6.1.3 CIM_EnabledLogicalElementCapabilities.MaxElementNameLen

399 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported
400 property has a value of TRUE.

7.6.2 Processor core state management

402 Processor core state management requires that the CIM_ProcessorCore.RequestStateChange( ) method
403 be supported (see 8.2) and the value of the CIM_ProcessorCore.RequestedState property not match 12
404 (Not Applicable).

7.6.2.1 Processor core state management support

406 When no CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_ProcessorCore
407 instance, processor core state management shall not be supported.

408 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_ProcessorCore
409 instance but the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported
410 property is an empty array, processor core state management shall not be supported.

411 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_ProcessorCore
412 instance and the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported
413 property is not an empty array, processor core state management shall be supported.

7.6.3 CIM_ProcessorCore.RequestedState

415 The CIM_ProcessorCore.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No
416 Change), or a value contained in the

417 CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property array of the associated
418 CIM_EnabledLogicalElementCapabilities instance (see 7.6.1.1).

419 When processor core state management is supported and the RequestStateChange( ) method is
420 successfully executed, the RequestedState property shall be set to the value of the RequestedState
421 parameter of the RequestStateChange( ) method. After the RequestStateChange( ) method has
422 successfully executed, the RequestedState and EnabledState properties shall have equal values with the
423 exception of the transitional requested state 11 (Reset). The value of the RequestedState property may
424 also change as a result of a request for a change to the processor's enabled state by a non-CIM
425 implementation.

426 7.6.3.1 RequestedState — 12 (Not Applicable) value

427 When processor core state management is not supported, the value of the
428 CIM_ProcessorCore.RequestedState property shall be 12 (Not Applicable).

429 7.6.3.2 RequestedState — 5 (No Change) value

430 When processor core state management is supported, the initial value of the
431 CIM_ProcessorCore.RequestedState property shall be 5 (No Change).

432 7.6.4 Modeling the current enabled state of the processor core

433 The current enabled state of the processor core is described by the
434 CIM_ProcessorCore.CoreEnabledState and CIM_ProcessorCore.EnabledState properties. Clauses
435 7.6.4.1 and 7.6.4.2 detail the requirements for implementing these two properties.

436 7.6.4.1 CIM_ProcessorCore.CoreEnabledState

437 Table 5 describes the mapping between the values of the CIM_ProcessorCore.CoreEnabledState
438 property and the corresponding description of the state of the processor core. The CoreEnabledState
439 property shall match the values that are specified in Table 5. When the RequestStateChange( ) method
440 executes but does not complete successfully, and the processor core is in an indeterminate state, the
441 CoreEnabledState property shall have a value of 0 (Unknown). The value of this property may also
442 change as a result of a change to the processor's enabled state by a non-CIM implementation.

443 **Table 5 – CIM_ProcessorCore.CoreEnabledState Value Descriptions**

Value	Description	Extended Description
0	Unknown	Processor core state is indeterminate, or the processor core state management is not supported.
2	Enabled	Processor core shall be enabled.
3	Disabled	Processor core shall be disabled.
4	Core Disabled User	Processor core shall be disabled through client configuration, which may occur through client invocation of RequestStateChange( ) or through a non-CIM implementation such as BIOS.
5	Core Disabled By Post Error	Processor core shall be disabled due to a POST error.

444 7.6.4.2 CIM_ProcessorCore.EnabledState

445 The CIM_ProcessorCore.EnabledState property shall be implemented in addition to the
446 CIM_ProcessorCore.CoreEnabledState property. When the CoreEnabledState property value matches a
447 value in the "CoreEnabledState Value" column in Table 6, the EnabledState property shall have the value
448 that matches the value in the "EnabledState Value" column in the same row in the table.

449   **Table 6 – Mapping for the CoreEnabledState property and EnabledState property values**

CoreEnabledState Value	Description	EnabledState Value	Description
0	Unknown	0 or 5	Unknown or Not Applicable
2	Enabled	2	Enabled
3	Disabled	3	Disabled
4	Core Disabled User	3	Disabled
5	Core Disabled By Post Error	3	Disabled

450   7.7   Modeling individual hardware threads

451   Modeling the individual hardware threads is optional functionality. When hardware threads are modeled,
452   the implementation shall model processor cores as described in 7.6 and shall instantiate an instance of
453   CIM_HardwareThread to represent each hardware thread. All the requirements in this clause and its
454   subclauses are applicable when an implementation instantiates the CIM_HardwareThread class.

455   The instance of CIM_HardwareThread shall be associated through an instance of
456   CIM_ConcreteComponent to only one instance of CIM_ProcessorCore that represents the processor core
457   that enables the hardware thread (Threading Processor Core).

458   For a given Host Processor, the number of instances of CIM_HardwareThread that are associated with
459   Threading Processor Cores, which in turn are associated with the Host Processor, shall be equal to the
460   value of the NumberOfHardwareThreads property of the instance of CIM_ProcessorCapabilities that is
461   associated with the Host Processor.

462   7.7.1   Hardware thread capabilities

463   When the CIM_EnabledLogicalElementCapabilities class is instantiated to represent the hardware thread
464   capabilities, the instance of CIM_EnabledLogicalElementCapabilities shall be associated with the
465   CIM_HardwareThread instance through an instance of CIM_ElementCapabilities and used for advertising
466   the capabilities of the CIM_HardwareThread instance.

467   At most one instance of CIM_EnabledLogicalElementCapabilities shall be associated with a given
468   instance of CIM_HardwareThread.

469   7.7.1.1   CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported

470   The RequestedStatesSupported property is an array that contains the supported requested states for the
471   instance of CIM_HardwareThread. This property shall be the super set of the values to be used as the
472   RequestedState parameter in the RequestStateChange( ) method (see 8.3). The value of the
473   RequestedStatesSupported property shall be an empty array or contain any combination of the following
474   values: 2 (Enabled), 3 (Disabled), or 11 (Reset).

475   7.7.1.2   CIM_EnabledLogicalElementCapabilities.ElementNameEditSupported

476   The ElementNameEditSupported property shall have a value of TRUE when the implementation supports
477   client modification of the CIM_HardwareThread.ElementName property.

478   7.7.1.3   CIM_EnabledLogicalElementCapabilities.MaxElementNameLen

479   The MaxElementNameLen property shall be implemented when the ElementNameEditSupported
480   property has a value of TRUE.

481 **7.7.2 Hardware thread state management**

482 Hardware thread state management requires that the CIM_HardwareThread.RequestStateChange( )
483 method be supported (see 8.3) and the value of the CIM_HardwareThread.RequestedState property not
484 match 12 (Not Applicable).

485 **7.7.2.1 Hardware thread state management support**

486 When no CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_HardwareThread
487 instance, hardware thread state management shall not be supported.

488 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_HardwareThread
489 instance but the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported
490 property is an empty array, hardware thread state management shall not be supported.

491 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_HardwareThread
492 instance and the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported
493 property is not an empty array, hardware thread state management shall be supported.

494 **7.7.3 CIM_HardwareThread.RequestedState**

495 The CIM_HardwareThread.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No
496 Change), or a value contained in the
497 CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property array of the associated
498 CIM_EnabledLogicalElementCapabilities instance (see 7.7.1.1).

499 When hardware thread state management is supported and the RequestStateChange( ) method is
500 successfully executed, the RequestedState property shall be set to the value of the RequestedState
501 parameter of the RequestStateChange( ) method. After the RequestStateChange( ) method has
502 successfully executed, the RequestedState and EnabledState properties shall have equal values with the
503 exception of the transitional requested state 11 (Reset). The value of the RequestedState property may
504 also change as a result of a request for a change to the hardware thread's enabled state by a non-CIM
505 implementation.

506 **7.7.3.1 RequestedState — 12 (Not Applicable) value**

507 When hardware thread state management is not supported, the value of the
508 CIM_HardwareThread.RequestedState property shall be 12 (Not Applicable).

509 **7.7.3.2 RequestedState — 5 (No Change) value**

510 When hardware thread state management is supported, the initial value of the
511 CIM_HardwareThread.RequestedState property shall be 5 (No Change).

512 **7.7.4 CIM_HardwareThread.EnabledState**

513 Table 7 describes the mapping between the values of the CIM_HardwareThread.EnabledState property
514 and the corresponding description of the state of the hardware thread. The EnabledState property shall
515 match the values that are specified in Table 7. When the RequestStateChange( ) method executes but
516 does not complete successfully, and the hardware thread is in an indeterminate state, the EnabledState
517 property shall have a value of 5 (Not Applicable). The value of this property may also change as a result
518 of a change to the hardware thread's enabled state by a non-CIM implementation.

519   **Table 7 – CIM_HardwareThread.EnabledState Value Descriptions**

Value	Description	Extended Description
2	Enabled	Hardware thread shall be enabled.
3	Disabled	Hardware thread shall be disabled.
5	Not Applicable	Hardware thread state is indeterminate, or hardware thread state management is not supported.

520 7.8   Modeling cache memory

521 Modeling the cache memory of the processor is optional. The implementation may instantiate instances of
522 CIM_Memory to represent the cache memory. All the requirements in this clause and its subclauses are
523 applicable when an implementation instantiates the CIM_Memory class that represents cache memory.

524 A single instance of CIM_Memory shall exist for each discrete cache memory. When the cache memory is
525 shared, the single instance of CIM_Memory shall be associated with multiple instances of CIM_Processor
526 or CIM_ProcessorCore. When the cache memory is not shared, the instance of CIM_Memory shall be
527 associated with exactly one instance of CIM_Processor or CIM_ProcessorCore.

528 When the optional behavior described in 7.6 is implemented, each instance of CIM_Memory that
529 represents the cache memory used by the processor core shall be associated with the instance of
530 CIM_ProcessorCore that represents the processor core through an instance of
531 CIM_AssociatedCacheMemory and shall not be associated with the Host Processor of the core.

532 When the optional behavior described in 7.6 is not implemented, each instance of CIM_Memory that
533 represents the cache memory used by the processor shall be associated through an instance of the
534 CIM_AssociatedCacheMemory to the instance of CIM_Processor.

535 7.8.1   Cache memory capabilities

536 When the CIM_EnabledLogicalElementCapabilities class is instantiated to represent the cache memory
537 capabilities, the instance of CIM_EnabledLogicalElementCapabilities shall be associated with the
538 CIM_Memory instance through an instance of CIM_ElementCapabilities and used for advertising the
539 capabilities of the CIM_Memory instance.

540 At most one instance of CIM_EnabledLogicalElementCapabilities shall be associated with a given
541 instance of CIM_Memory.

542 7.8.1.1   CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported

543 The RequestedStatesSupported property is an array that contains the supported requested states for the
544 instance of CIM_Memory. This property shall be the super set of the values to be used as the
545 RequestedState parameter in the RequestStateChange( ) method (see 8.4). The value of the
546 RequestedStatesSupported property shall be an empty array or contain any combination of the following
547 values: 2 (Enabled), 3 (Disabled), or 11 (Reset).

548 7.8.1.2   CIM_EnabledLogicalElementCapabilities.ElementNameEditSupported

549 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports
550 client modification of the CIM_Memory.ElementName property.

551 7.8.1.3   CIM_EnabledLogicalElementCapabilities.MaxElementNameLen

552 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported
553 property has a value of TRUE.

554 **7.8.2 Cache memory state management**

555 Cache memory state management requires that the CIM_Memory.RequestStateChange( ) method be
556 supported (see 8.4) and the value of the CIM_Memory.RequestedState property not match 12 (Not
557 Applicable).

558 **7.8.2.1 Cache memory state management support**

559 When no CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_Memory instance,
560 cache memory state management shall not be supported.

561 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_Memory instance
562 but the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property is an
563 empty array, cache memory state management shall not be supported.

564 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_Memory instance
565 and the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property is not
566 an empty array, cache memory state management shall be supported.

567 **7.8.3 CIM_Memory.RequestedState**

568 The CIM_Memory.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No Change),
569 or a value contained in the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property
570 array of the associated CIM_EnabledLogicalElementCapabilities instance (see 7.8.1.1).

571 When cache memory state management is supported and the RequestStateChange( ) method is
572 successfully executed, the RequestedState property shall be set to the value of the RequestedState
573 parameter of the RequestStateChange( ) method. After the RequestStateChange( ) method has
574 successfully executed, the RequestedState and EnabledState properties shall have equal values with the
575 exception of the transitional requested state 11 (Reset). The value of the RequestedState property may
576 also change as a result of a request for a change to the cache memory's enabled state by a non-CIM
577 implementation.

578 **7.8.3.1 RequestedState — 12 (Not Applicable) value**

579 When cache memory state management is not supported, the value of the CIM_Memory.RequestedState
580 property shall be 12 (Not Applicable).

581 **7.8.3.2 RequestedState — 5 (No Change) value**

582 When cache memory state management is supported, the initial value of the
583 CIM_Memory.RequestedState property shall be 5 (No Change).

584 **7.8.4 CIM_Memory.EnabledState**

585 Table 8 describes the mapping between the values of the CIM_Memory.EnabledState property and the
586 corresponding description of the state of the cache memory. The EnabledState property shall match the
587 values that are specified in Table 8. When the RequestStateChange( ) method executes but does not
588 complete successfully, and the cache memory is in an indeterminate state, the EnabledState property
589 shall have value of 5 (Not Applicable). The value of this property may also change as a result of a change
590 to the cache memory's enabled state by a non-CIM implementation.

591 **Table 8 – CIM_Memory.EnabledState value descriptions**

Value	Description	Extended Description
2	Enabled	Cache memory shall be enabled.
3	Disabled	Cache memory shall be disabled.
5	Not Applicable	Cache memory state is indeterminate, or cache memory state management is not supported.

592 7.9 Modeling physical aspects of processor and cache memory

593 The *Physical Asset Profile* may be implemented to model the physical aspects of a processor, including
594 the asset information of the processor or the internal or off-chip cache memory.

595 When the processor's or internal cache memory's physical aspects are represented, a CIM_Chip instance
596 shall be instantiated and associated with the instance of CIM_Processor or with any instances of
597 CIM_Memory that represent the internal cache through instances of CIM_Realizes.

598 When the off-chip cache memory is represented along with its physical aspects, a CIM_PhysicalMemory
599 instance shall be instantiated and associated with the instance of CIM_Memory through an instance of
600 CIM_Realizes.

601 When processor cores or hardware threads for the processor are modeled with the physical aspects of
602 the processor, the instances of CIM_ProcessorCore and CIM_HardwareThread shall not be associated
603 with the instance of CIM_Chip that represents the physical aspects of the processor.

604 The configuration capacity of the managed system for processors may be modeled using the optional
605 behavior specified in the "Modeling Configuration Capacity" clause of the *Physical Asset Profile*.

606 8 Methods

607 This clause details the requirements for supporting intrinsic operations and extrinsic methods for the CIM
608 elements defined by this profile.

609 8.1 CIM_Processor.RequestStateChange( )

610 Invocation of the CIM_Processor.RequestStateChange( ) method changes the element's state to the
611 value that is specified in the RequestedState parameter.

612 Return code values for the RequestStateChange( ) method shall be as specified in Table 9. Parameters
613 of the RequestStateChange( ) method are specified in Table 10.

614 When processor state management is supported, the RequestStateChange( ) method shall be
615 implemented and shall not return a value of 1 (Not Supported) (see 7.3.1).

616 Invoking the CIM_Processor.RequestStateChange( ) method multiple times could result in earlier
617 requests being overwritten or lost.

618 No standard messages are defined for this method.

619 **Table 9 – CIM_Processor.RequestStateChange( ) method: Return code values**

Value	Description
0	Request was successfully executed.
1	Method is not supported in the implementation.
2	Error occurred
4096	Job started

620 **Table 10 – CIM_Processor.RequestStateChange( ) method: Parameters**

Qualifiers	Name	Type	Description/Values
IN, REQ	RequestedState	uint16	Valid state values: 2 (Enabled) 3 (Disabled) 11 (Reset)
OUT	Job	CIM_ConcreteJob REF	Returned if job started
IN, REQ	TimeoutPeriod	datetime	Client-specified maximum amount of time the transition to a new state is supposed to take: 0 or NULL – No time requirements <interval> – Maximum time allowed

621 **8.2 CIM_ProcessorCore.RequestStateChange( )**

622 Invocation of the CIM_ProcessorCore.RequestStateChange( ) method changes the element's state to the
623 value that is specified in the RequestedState parameter.

624 Return code values for the RequestStateChange( ) method shall be as specified in Table 11. Parameters
625 of the RequestStateChange( ) method are specified in Table 12.

626 When processor core state management is supported, the RequestStateChange( ) method shall be
627 implemented and shall not return a value of 1 (Not Supported) (see 7.6.2.1).

628 Invoking the CIM_ProcessorCore.RequestStateChange( ) method multiple times could result in earlier
629 requests being overwritten or lost.

630 No standard messages are defined for this method.

631 **Table 11 – CIM_ProcessorCore.RequestStateChange( ) method: Return code values**

Value	Description
0	Request was successfully executed.
1	Method is not supported in the implementation.
2	Error occurred
4096	Job started

632 **Table 12 – CIM_ProcessorCore.RequestStateChange( ) method: Parameters**

Qualifiers	Name	Type	Description/Values
IN, REQ	RequestedState	uint16	Valid state values: 2 (Enabled) 3 (Disabled) 11 (Reset)
OUT	Job	CIM_ConcreteJob REF	Returned if job started
IN, REQ	TimeoutPeriod	datetime	Client-specified maximum amount of time the transition to a new state is supposed to take: 0 or NULL – No time requirements <interval> – Maximum time allowed

8.3 CIM_HardwareThread.RequestStateChange( )

634 Invocation of the CIM_HardwareThread.RequestStateChange( ) method changes the element's state to
635 the value that is specified in the RequestedState parameter.

636 Return code values for the RequestStateChange( ) method shall be as specified in Table 13. Parameters
637 of the RequestStateChange( ) method are specified in Table 14.

638 When hardware thread state management is supported, the RequestStateChange( ) method shall be
639 implemented and shall not return a value of 1 (Not Supported) (see 7.7.2.1).

640 Invoking the CIM_HardwareThread.RequestStateChange( ) method multiple times could result in earlier
641 requests being overwritten or lost.

642 No standard messages are defined for this method.

643 **Table 13 – CIM_HardwareThread.RequestStateChange( ) method: Return code values**

Value	Description
0	Request was successfully executed.
1	Method is not supported in the implementation.
2	Error occurred
4096	Job started

644 **Table 14 – CIM_HardwareThread.RequestStateChange( ) method: Parameters**

Qualifiers	Name	Type	Description/Values
IN, REQ	RequestedState	uint16	Valid state values: 2 (Enabled) 3 (Disabled) 11 (Reset)
OUT	Job	CIM_ConcreteJob REF	Returned if job started
IN, REQ	TimeoutPeriod	datetime	Client-specified maximum amount of time the transition to a new state is supposed to take: 0 or NULL – No time requirements <interval> – Maximum time allowed

645 **8.4 CIM_Memory.RequestStateChange( )**

646 Invocation of the CIM_Memory.RequestStateChange( ) method changes the element's state to the value
647 that is specified in the RequestedState parameter.

648 Return code values for the RequestStateChange( ) method shall be as specified in Table 15. Parameters
649 of the RequestStateChange( ) method are specified in Table 16.

650 When memory state management is supported, the RequestStateChange( ) method shall be implemented
651 and shall not return a value of 1 (Not Supported) (see 7.8.2.1).

652 Invoking the CIM_Memory.RequestStateChange( ) method multiple times could result in earlier requests
653 being overwritten or lost.

654 No standard messages are defined for this method.

655 **Table 15 – CIM_Memory.RequestStateChange( ) method: Return code values**

Value	Description
0	Request was successfully executed.
1	Method is not supported in the implementation.
2	Error occurred
4096	Job started

656 **Table 16 – CIM_Memory.RequestStateChange( ) method: Parameters**

Qualifiers	Name	Type	Description/Values
IN, REQ	RequestedState	uint16	Valid state values: 2 (Enabled) 3 (Disabled) 11 (Reset)
OUT	Job	CIM_ConcreteJob REF	Returned if job started
IN, REQ	TimeoutPeriod	datetime	Client-specified maximum amount of time the transition to a new state is supposed to take: 0 or NULL – No time requirements <interval> – Maximum time allowed

657 **8.5 Profile conventions for operations**

658 This profile specification defines operations in terms of DSP0200.

659 For each profile class (including associations), the implementation requirements for operations, including
660 those in the following default list, are specified in class-specific subclauses of this clause.

661 The default list of operations is as follows:

662 • Associators( )

663 • AssociatorNames( )

664 • EnumerateInstances( )

665   •   EnumerateInstanceNames( )

666   •   GetInstance( )

667   •   References( )

668   •   ReferenceNames( )

669   8.6   CIM_AssociatedCacheMemory

670 Table 17 lists implementation requirements for operations. If implemented, these operations shall be
671 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 17, all operations
672 in the default list in 8.5 shall be implemented as defined in DSP0200.

673   NOTE   Related profiles may define additional requirements on operations for the profile class.

674   **Table 17 – Operations: CIM_AssociatedCacheMemory**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

675   8.7   CIM_ConcreteComponent — References CIM_HardwareThread and
676   CIM_Processor

677 Table 18 lists implementation requirements for operations. If implemented, these operations shall be
678 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 18, all operations
679 in the default list in 8.5 shall be implemented as defined in DSP0200.

680   NOTE   Related profiles may define additional requirements on operations for the profile class.

681   **Table 18 – Operations: CIM_ConcreteComponent**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

682   8.8   CIM_ConcreteComponent — References CIM_ProcessorCore and
683   CIM_Processor

684 Table 19 lists implementation requirements for operations. If implemented, these operations shall be
685 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 19, all operations
686 in the default list in 8.5 shall be implemented as defined in DSP0200.

687   NOTE   Related profiles may define additional requirements on operations for the profile class.

688   **Table 19 – Operations: CIM_ConcreteComponent**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

689   **8.9   CIM_ElementCapabilities — References CIM_HardwareThread and**
690   **CIM_EnabledLogicalElementCapabilities**

691   Table 20 lists implementation requirements for operations. If implemented, these operations shall be
692   implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 20, all operations
693   in the default list in 8.5 shall be implemented as defined in DSP0200.

694   NOTE   Related profiles may define additional requirements on operations for the profile class.

695   **Table 20 – Operations: CIM_ElementCapabilities**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

696   **8.10 CIM_ElementCapabilities — References CIM_Memory and**
697   **CIM_EnabledLogicalElementCapabilities**

698   Table 21 lists implementation requirements for operations. If implemented, these operations shall be
699   implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 21, all operations
700   in the default list in 8.5 shall be implemented as defined in DSP0200.

701   NOTE   Related profiles may define additional requirements on operations for the profile class.

702   **Table 21 – Operations: CIM_ElementCapabilities**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

703   **8.11 CIM_ElementCapabilities — References CIM_Processor and**
704   **CIM_ProcessorCapabilities**

705   Table 22 lists implementation requirements for operations. If implemented, these operations shall be
706   implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 22, all operations
707   in the default list in 8.5 shall be implemented as defined in DSP0200.

708   NOTE   Related profiles may define additional requirements on operations for the profile class.

709 **Table 22 – Operations: CIM_ElementCapabilities**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

710 8.12 CIM_ElementCapabilities — References CIM_ProcessorCore and
711 CIM_EnabledLogicalElementCapabilities

712 Table 23 lists implementation requirements for operations. If implemented, these operations shall be
713 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 23, all operations
714 in the default list in 8.5 shall be implemented as defined in DSP0200.

715 NOTE Related profiles may define additional requirements on operations for the profile class.

716 **Table 23 – Operations: CIM_ElementCapabilities**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

717 8.13 CIM_EnabledLogicalElementCapabilities

718 All operations in the default list in 8.5 shall be implemented as defined in DSP0200.

719 NOTE Related profiles may define additional requirements on operations for the profile class.

720 8.14 CIM_HardwareThread

721 Table 24 lists implementation requirements for operations. If implemented, these operations shall be
722 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 24, all operations
723 in the default list in 8.5 shall be implemented as defined in DSP0200.

724 NOTE Related profiles may define additional requirements on operations for the profile class.

725 **Table 24 – Operations: CIM_HardwareThread**

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.14.1.	None

726 8.14.1 CIM_HardwareThread — ModifyInstance

727 This clause details the requirements for the ModifyInstance operation applied to an instance of
728 CIM_HardwareThread. The ModifyInstance operation may be supported.

729 The ModifyInstance operation shall be supported and the CIM_HardwareThread.ElementName property
730 shall be modifiable when the ElementNameEditSupported property of the

731 CIM_EnabledLogicalElementCapabilities instance that is associated with the CIM_HardwareThread
732 instance has a value of TRUE. See 7.7.1.2.

8.15 CIM_Memory

734 Table 25 lists implementation requirements for operations. If implemented, these operations shall be
735 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 25, all operations
736 in the default list in 8.5 shall be implemented as defined in DSP0200.

737 NOTE   Related profiles may define additional requirements on operations for the profile class.

738 **Table 25 – Operations: CIM_Memory**

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.15.1.	None

8.15.1 CIM_Memory — ModifyInstance

740 This clause details the requirements for the ModifyInstance operation applied to an instance of
741 CIM_Memory. The ModifyInstance operation may be supported.

742 The ModifyInstance operation shall be supported and the CIM_Memory.ElementName property shall be
743 modifiable when the ElementNameEditSupported property of the
744 CIM_EnabledLogicalElementCapabilities instance that is associated with the CIM_Memory instance has
745 a value of TRUE. See clause 7.8.1.2.

8.16 CIM_Processor

747 Table 26 lists implementation requirements for operations. If implemented, these operations shall be
748 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 26, all operations
749 in the default list in 8.5 shall be implemented as defined in DSP0200.

750 NOTE   Related profiles may define additional requirements on operations for the profile class.

751 **Table 26 – Operations: CIM_Processor**

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.16.1.	None

8.16.1 CIM_Processor — ModifyInstance

753 This clause details the requirements for the ModifyInstance operation applied to an instance of
754 CIM_Processor. The ModifyInstance operation may be supported.

755 The ModifyInstance operation shall be supported and the CIM_Processor.ElementName property shall be
756 modifiable when the ElementNameEditSupported property of the
757 CIM_EnabledLogicalElementCapabilities instance that is associated with the CIM_Processor instance
758 has a value of TRUE. See 7.2.4.

8.17 CIM_ProcessorCapabilities

760 All operations in the default list in 8.5 shall be implemented as defined in DSP0200.

761 NOTE   Related profiles may define additional requirements on operations for the profile class.

762 **8.18 CIM_ProcessorCore**

763 Table 27 lists implementation requirements for operations. If implemented, these operations shall be
764 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 27, all operations
765 in the default list in 8.5 shall be implemented as defined in DSP0200.

766 NOTE   Related profiles may define additional requirements on operations for the profile class.

767 **Table 27 – Operations: CIM_ProcessorCore**

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.18.1.	None

768 **8.18.1 CIM_ProcessorCore — ModifyInstance**

769 This clause details the requirements for the ModifyInstance operation applied to an instance of
770 CIM_ProcessorCore. The ModifyInstance operation may be supported.

771 The ModifyInstance operation shall be supported and the CIM_ProcessorCore.ElementName property
772 shall be modifiable when the ElementNameEditSupported property of the
773 CIM_EnabledLogicalElementCapabilities instance that is associated with the CIM_ProcessorCore
774 instance has a value of TRUE. See 7.6.1.2.

775 **8.19 CIM_SystemDevice**

776 Table 28 lists implementation requirements for operations. If implemented, these operations shall be
777 implemented as defined in DSP0200. In addition, and unless otherwise stated in Table 28, all operations
778 in the default list in 8.5 shall be implemented as defined in DSP0200.

779 NOTE   Related profiles may define additional requirements on operations for the profile class.

780 **Table 28 – Operations: CIM_SystemDevice**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

781 # 9   Use cases

782 This clause contains object diagrams and use cases for the *CPU Profile*.

783 ## 9.1   Object diagrams

784 Figure 2 represents a possible instantiation of the *CPU Profile*. In this instantiation, cpu1 belongs to
785 system1. The capabilities of cpu1 are represented with capabilities1. cpu1 has cache memory
786 represented by memory1. The *CPU Profile* implementation and versioning information is advertised
787 through profile2.
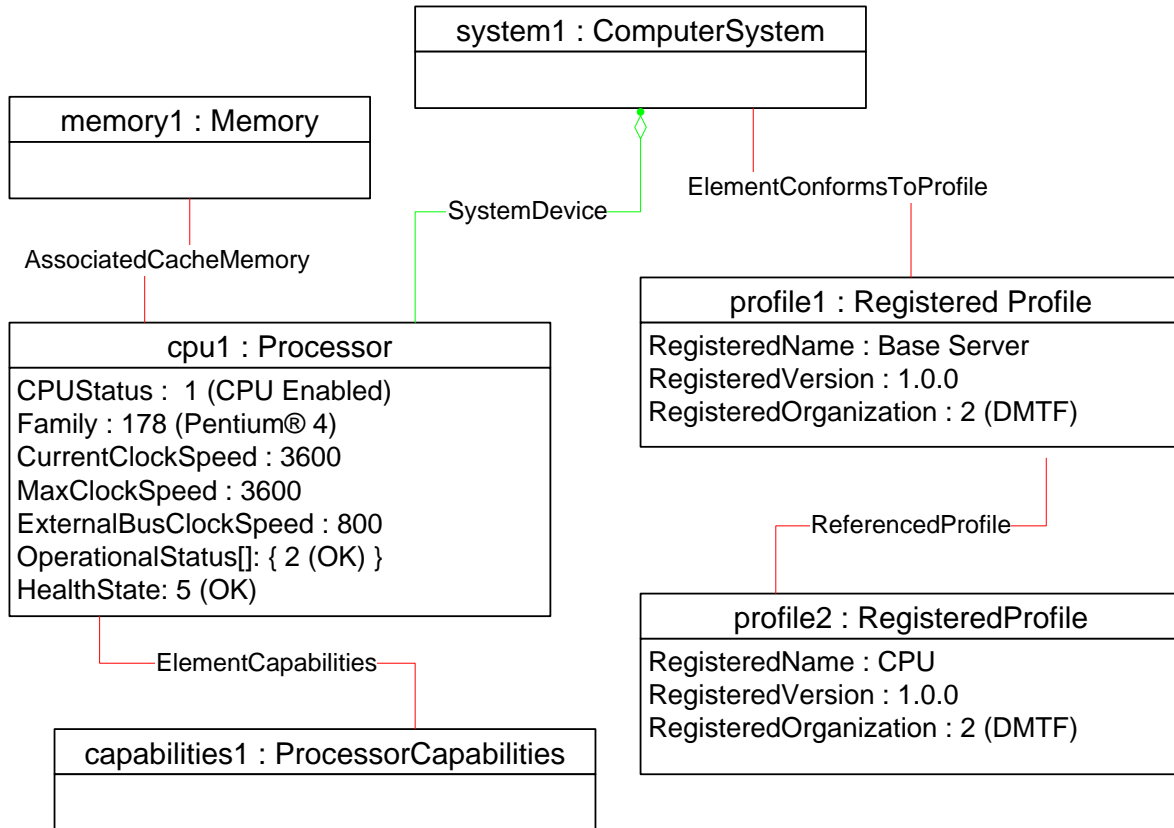
788

789 **Figure 2 – Registered Profile**

790   Figure 3 represents a possible instantiation of the *CPU Profile* representing a dual core processor, cpu1.
791   The individual cores and hardware threads of cpu1 are not represented, but capabilities1 advertises that
792   cpu1 is a dual core processor capable of two hardware threads, one thread per each core. If system1
793   supports *SMBIOS Reference Specification* 2.6 or later, the value of the NumberOfProcessorCores
794   property will be equal to the SMBIOS Processor Information structure's Core Count structure value, and
795   the value of the NumberOfHardwareThreads property will be equal to the SMBIOS Processor Information
796   structure's Thread Count structure value. memory1 and memory2 are the cache memories of cpu1.
797   Memory1 represents the level 1 cache, and memory2 is the level 2 cache, as denoted by the instances of
798   CIM_AssociatedCacheMemory that associate memory1 and memory2 with cpu1. The physical aspects of
799   cpu1 are represented by chip1, associated to cpu1 through an instance of CIM_Realizes.
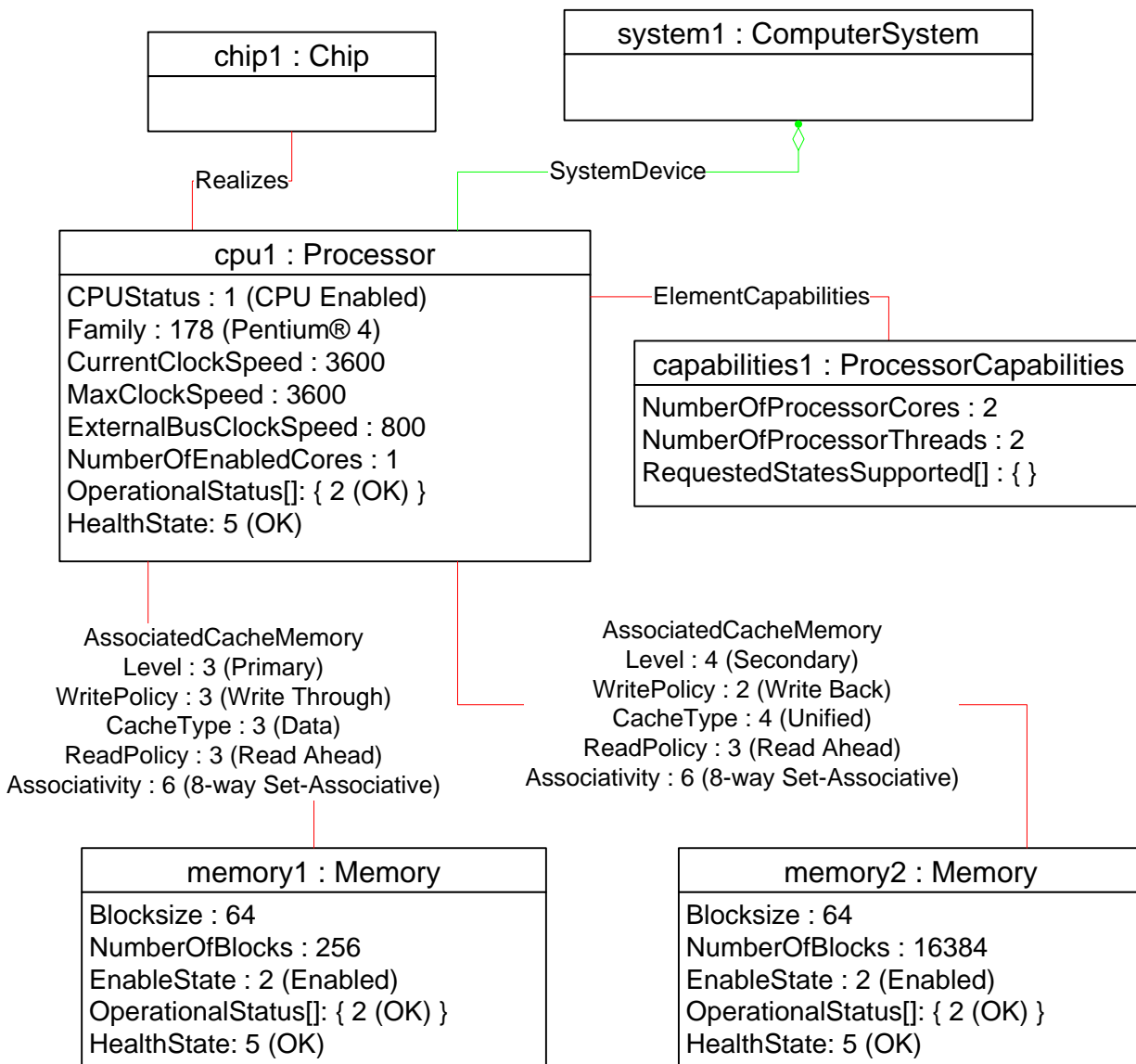
800

801   **Figure 3 – Multi-core CPU**

802   Figure 4 represents a possible instantiation of the *CPU Profile* representing a dual core processor, cpu1.
803   Each of the processor cores is represented by an instance of CIM_ProcessorCore: core1 and core2,
804   associated to the Host Processor, cpu1, through instances of CIM_ConcreteComponent. Each of the
805   cores has one hardware thread, represented by thread1 and thread2, associated with it through instances
806   of CIM_ConcreteComponent. The cache memories, memory1 and memory2, are associated to the
807   processor cores that use them. Based on the capabilities of core1 and core2, represented by
808   capabilities2, both processor cores can be enabled or disabled using the RequestStateChange( ) method.
809   Figure 5 shows the same instantiation of *CPU Profile* after the RequestStateChange( ) method on core2
810   has successfully executed.
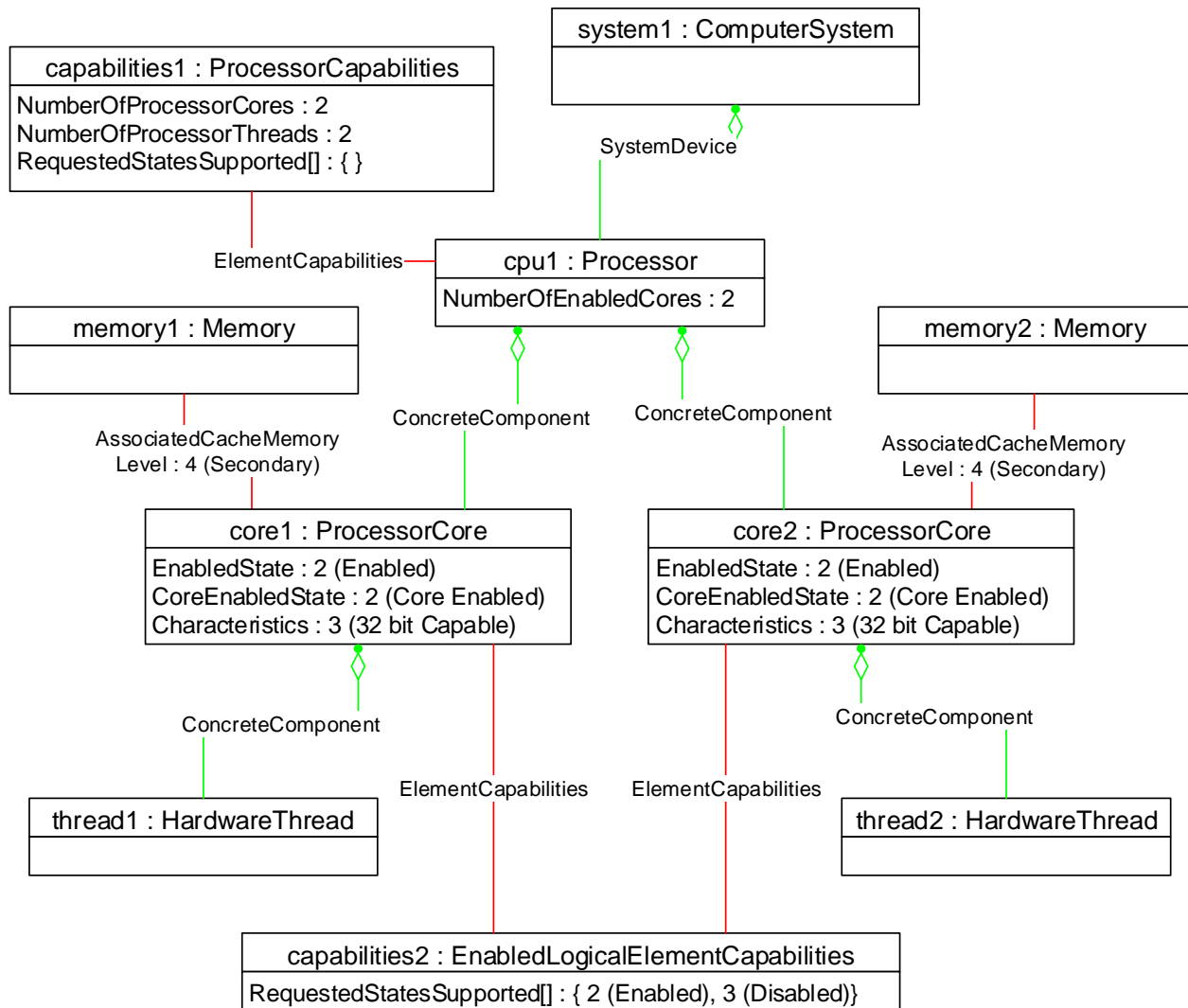
811

812   **Figure 4 – Detailed multi-core CPU**

813   Figure 5 represents a possible instantiation of the *CPU Profile* in which one of the cores of a dual core
814   processor, cpu1, has been disabled by the user using the RequestStateChange( ) method. core2's
815   EnabledState property has value of 3 (Disabled) and the CoreEnabledState property has value 4 (Core
816   Disabled by User).
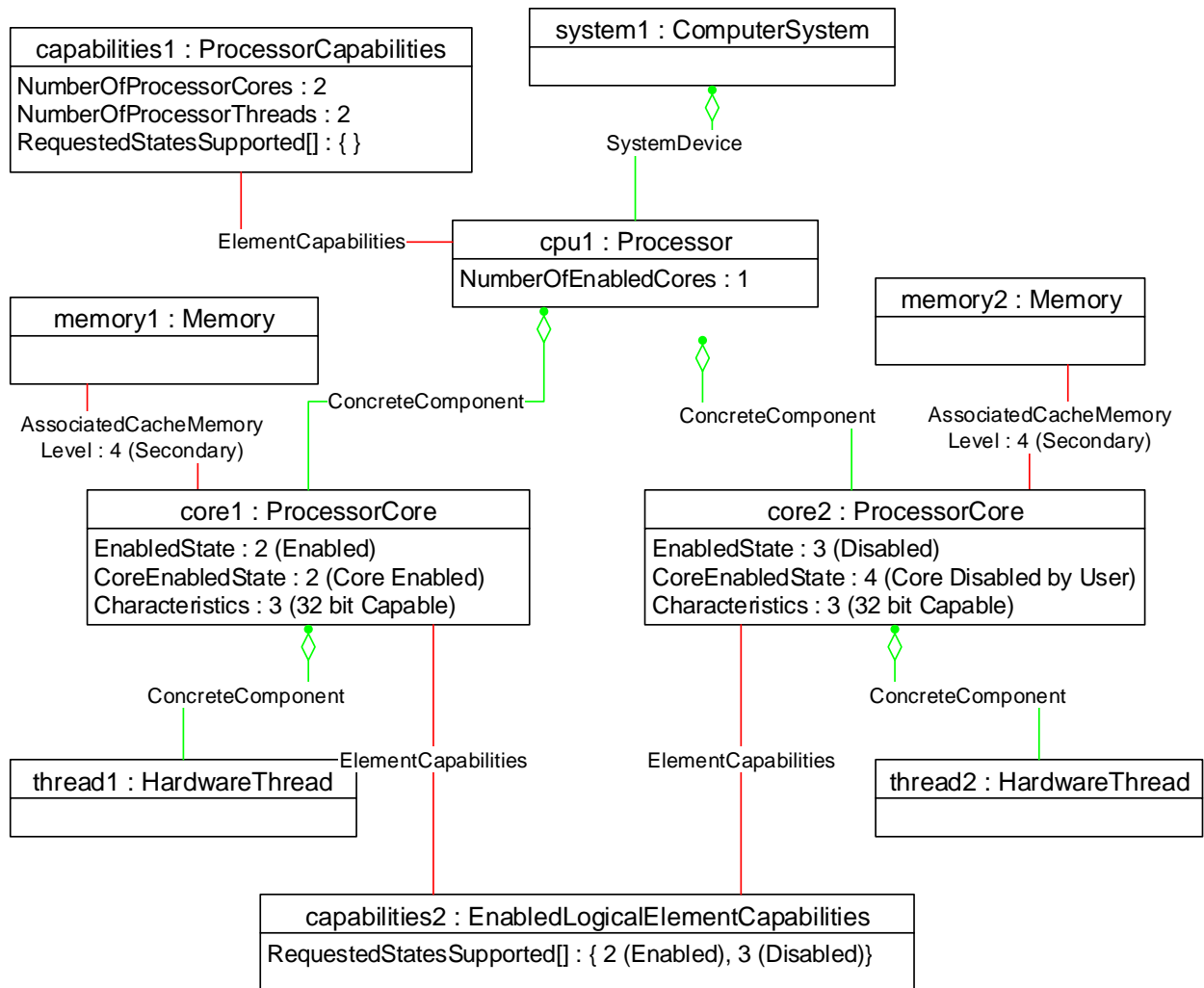
817

818   **Figure 5 – Multi-core CPU with a disabled core**

819   Figure 6 represents a possible instantiation of the *CPU Profile* representing a single core processor with
820   multiple threads. thread1 and thread2 represent the hardware threads that exist on core1 and are
821   associated to core1 through an instance of CIM_ConcreteComponent. cpu1 advertises the capabilities of
822   multiple hardware threads through the capabilities1 NumberOfProcessorThreads property. The cache
823   memory, memory1, is associated to core1, which is using the cache memory.
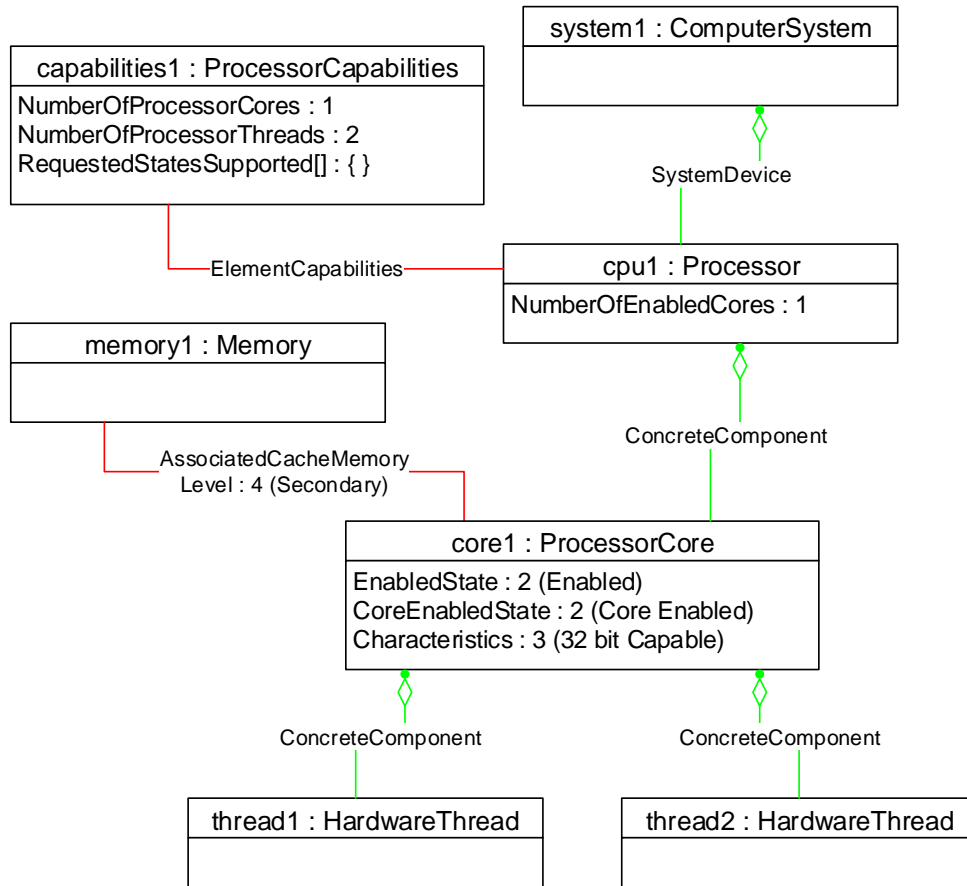
824

825   **Figure 6 – Single Core, Multi-Hardware Thread CPU**

826   Figure 7 represents another instantiation of the *CPU Profile*. In this case, cpu1's cache memory,
827   memory1, has a separate package represented by pmem1 and associated to memory1 through an
828   instance of CIM_Realizes. The existence of pmem1 associated with the cpu1's cache memory indicates
829   that the processor uses off-chip cache memory.
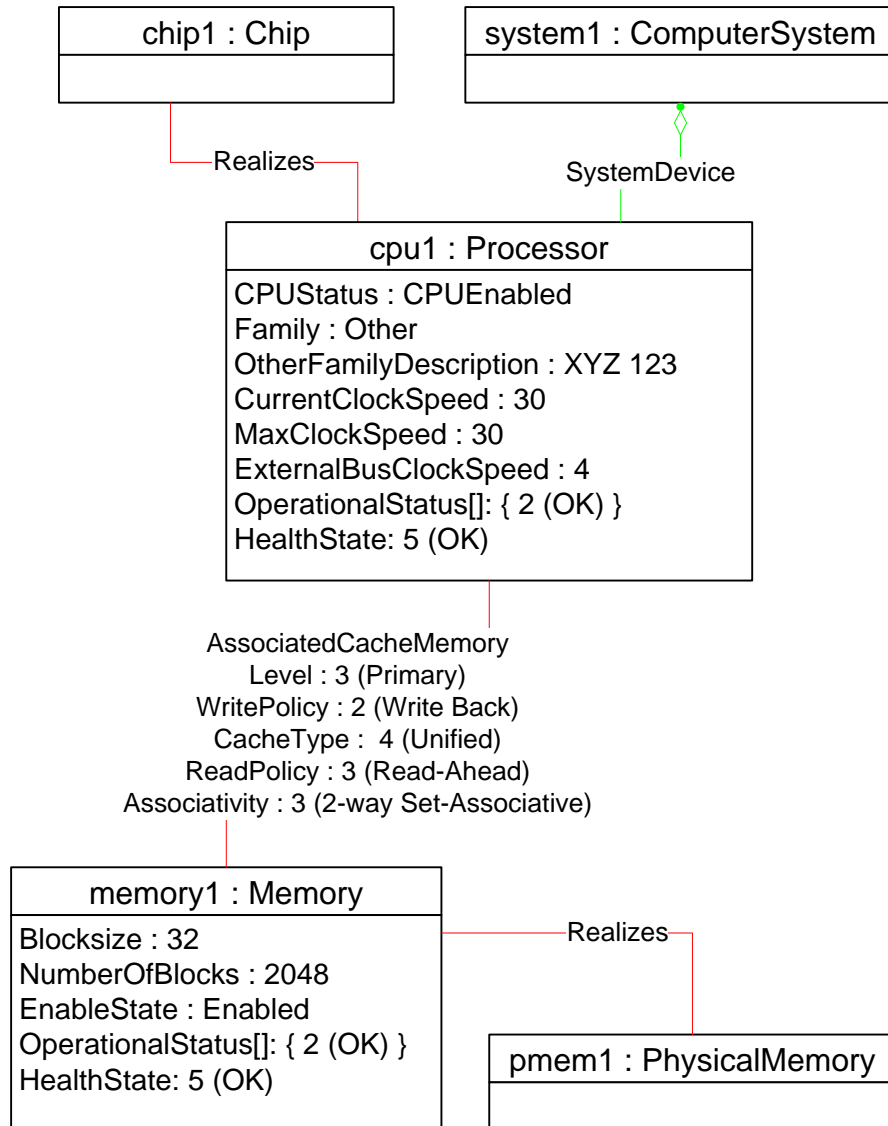
830

831   **Figure 7 – Processor with Off-Chip Cache**

832   ## 9.2   Change the enabled state of the memory to the desired state

833   A client can change the enabled state of the memory as follows:

834   1)   Select the instance of CIM_Memory.

835   2)   Select the associated instance of CIM_EnabledLogicalElementCapabilities and verify whether
836   the RequestedStatesSupported property contains the desired state.

837   3)   If the RequestedStatesSupported property contains the desired state, select the instance of
838   CIM_Memory and execute the RequestStateChange( ) method with the desired state as a
839   RequestedState parameter.

840   After the successful execution of the method, the EnabledState property of the instance of CIM_Memory
841   will have the value of the desired state.

842   ## 9.3   Change the enabled state of the CPU to the desired state

843   A client can change the enabled state of the CPU as follows:

844   1)   Select the instance of CIM_Processor.

845   2)   Select the associated instance of CIM_ProcessorCapabilities and verify whether the
846   RequestedStatesSupported property contains the desired state.

847   3)   If the RequestedStatesSupported property contains the desired state, select the instance of
848   CIM_Processor and execute the RequestStateChange( ) method with the desired state as a
849   RequestedState parameter.

850   After the successful execution of the method, the EnabledState property of the instance of
851   CIM_Processor will have the value of the desired state.

852   ## 9.4   Change the enabled state of the CPU's core to the desired state

853   A client can change the enabled state of the CPU's core as follows:

854   1)   Select the instance of CIM_ProcessorCore.

855   2)   Select the associated instance of CIM_EnabledLogicalElementCapabilities and verify whether
856   the RequestedStatesSupported property contains the desired state.

857   3)   If the RequestedStatesSupported property contains the desired state, select the instance of
858   CIM_ProcessorCore and execute the RequestStateChange( ) method with the desired state as
859   a RequestedState parameter.

860   After the successful execution of the method, the EnabledState property of the instance of
861   CIM_ProcessorCore will have the value of the desired state.

862   ## 9.5   Change the enabled state of the CPU's hardware thread to the desired state

863   A client can change the enabled state of the CPU's hardware thread as follows:

864   1)   Select the instance of CIM_HardwareThread.

865   2)   Select the associated instance of CIM_EnabledLogicalElementCapabilities and verify whether
866   the RequestedStatesSupported property contains the desired state.

867   3)   If the RequestedStatesSupported property contains the desired state, select the instance of
868   CIM_ProcessorThread and execute the RequestStateChange( ) method with the desired state
869   as a RequestedState parameter.

870 After the successful execution of the method, the EnabledState property of the instance of
871 CIM_HardwareThread will have the value of the desired state.

9.6 Retrieve all the processor cores for the CPU

873 A client can retrieve all of the processor cores for the CPU by selecting all the CIM_ProcessorCore
874 instances that are associated with the given instance of CIM_Processor through instances of
875 CIM_Component.

9.7 Retrieve all the hardware threads for the CPU

877 A client can retrieve all of the hardware threads for the CPU as follows:

878　　1)　Select all the CIM_ProcessorCore instances that are associated with the given instance of
879　　　　CIM_Processor through instances of CIM_Component.

880　　2)　For each instance of CIM_ProcessorCore, select the instances of CIM_HardwareThread that
881　　　　are associated through instances of CIM_Component.

9.8 Retrieve CPU's cache memory information for the CPU

883 A client can retrieve the CPU's cache memory information as follows:

884　　1)　Select all the instances of CIM_ProcessorCore that are associated with the given instance of
885　　　　CIM_Processor through instances of CIM_Component.

886　　2)　If no instance of CIM_ProcessorCore exists, select the instances of
887　　　　CIM_AssociatedCacheMemory that reference the given instance of CIM_Processor, as well as
888　　　　all the instances of CIM_Memory that are associated with the given instance of CIM_Processor
889　　　　through instances of CIM_AssociatedCacheMemory.

890　　3)　Otherwise, for each instance of CIM_ProcessorCore, select the instances of
891　　　　CIM_AssociatedCacheMemory that reference the instance of CIM_ProcessorCore, as well as
892　　　　all the instances of CIM_Memory that are associated with the instance of CIM_ProcessorCore
893　　　　through instances of CIM_AssociatedCacheMemory.

10 CIM Elements

895 Table 29 shows the instances of CIM Elements for this profile. Instances of the CIM Elements shall be
896 implemented as described in Table 29. Clauses 7 ("Implementation") and 8 ("Methods") may impose
897 additional requirements on these elements.

898　　　　　　　　　　　　　**Table 29 – CIM Elements: CPU Profile**

Element Name	Requirement	Description
Classes		
CIM_AssociatedCacheMemory	Optional	See 10.1 and 7.8.
CIM_ConcreteComponent (references CIM_HardwareThread and CIM_ProcessorCore)	Optional	See 10.2.
CIM_ConcreteComponent (references CIM_ProcessorCore and CIM_Processor)	Optional	See 10.3.

Element Name	Requirement	Description
CIM_ElementCapabilities (references CIM_HardwareThread and CIM_EnabledLogicalElementCapabilities)	Optional	See 10.4.
CIM_ElementCapabilities (references CIM_Memory and CIM_EnabledLogicalElementCapabilities)	Optional	See 10.5.
CIM_ElementCapabilities (references CIM_Processor and CIM_ProcessorCapabilities)	Optional	See 10.6.
CIM_ElementCapabilities (references CIM_ProcessorCore and CIM_EnabledLogicalElementCapabilities)	Optional	See 10.7.
CIM_EnabledLogicalElementCapabilities	Optional	See 7.6.1, 7.7.1, 7.8.1, and 10.7.
CIM_HardwareThread	Optional	See 10.9.
CIM_Memory	Optional	See 10.10 and 7.8.
CIM_Processor	Mandatory	See 7.1 and 10.11.
CIM_ProcessorCapabilities	Optional	See 7.2 and 10.12.
CIM_ProcessorCore	Optional	See 10.13.
CIM_RegisteredProfile	Mandatory	See 10.14.
CIM_SystemDevice	Mandatory	See 10.15.
Indications		
None defined in this profile		

899   **10.1 CIM_AssociatedCacheMemory**

900   CIM_AssociatedCacheMemory associates an instance of CIM_Processor or CIM_ProcessorCore with an
901   instance of CIM_Memory that represents the cache memory of the processor. Table 30 contains the
902   requirements for elements of this class.

903   **Table 30 – Class: CIM_AssociatedCacheMemory**

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** This property shall reference the instance of CIM_Memory that represents the cache memory.
Dependent	Mandatory	**Key:** This property shall reference the instance of CIM_Processor or CIM_ProcessorCore. See 7.8 for more details.
Level	Mandatory	None
WritePolicy	Mandatory	None
CacheType	Mandatory	None
ReadPolicy	Mandatory	None
Associativity	Mandatory	None
OtherLevelDescription	Conditional	This property shall be implemented when the Level property has a value of 1 (Other).
OtherWritePolicyDescription	Conditional	This property shall be implemented when the WritePolicy property has a value of 1 (Other).
OtherCacheTypeDescription	Conditional	This property shall be implemented when the CacheType property has a value of 1 (Other).

904   **10.2 CIM_ConcreteComponent — References CIM_HardwareThread and**
905   **CIM_ProcessorCore**

906   CIM_ConcreteComponent associates an instance of CIM_ProcessorCore (the Threading Processor Core)
907   with an instance CIM_HardwareThread that represents a hardware thread. CIM_ConcreteComponent
908   shall be instantiated when the Threading Processor Core and the instance of CIM_HardwareThread are
909   instantiated. Table 31 contains the requirements for elements of this class.

910   **Table 31 – Class: CIM_ConcreteComponent — References CIM_HardwareThread and**
911   **CIM_ProcessorCore**

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** This property shall reference the Threading Processor Core.
PartComponent	Mandatory	**Key:** This property shall reference the CIM_HardwareThread that represents the hardware thread.

912 **10.3 CIM_ConcreteComponent — References CIM_ProcessorCore and**
913 **CIM_Processor**

914 CIM_ConcreteComponent associates an instance of CIM_Processor (the Host Processor) with an
915 instance CIM_ProcessorCore that represents a processor core. CIM_ConcreteComponent shall be
916 instantiated when the Host Processor and the instance of CIM_ProcessorCore are instantiated. Table 32
917 contains the requirements for elements of this class.

918 **Table 32 – Class: CIM_ConcreteComponent — References CIM_ProcessorCore and**
919 **CIM_Processor**

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** This property shall reference the Host Processor.
PartComponent	Mandatory	**Key:** This property shall reference the CIM_ProcessorCore that represents the hosted processor cores.

920 **10.4 CIM_ElementCapabilities — References CIM_HardwareThread and**
921 **CIM_EnabledLogicalElementCapabilities**

922 CIM_ElementCapabilities associates an instance of CIM_HardwareThread with the instance of
923 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of
924 CIM_HardwareThread.

925 CIM_ElementCapabilities is mandatory when the instance of CIM_HardwareThread and the instance of
926 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of
927 CIM_HardwareThread exist. Table 33 contains the requirements for elements of this class.

928 **Table 33 – Class: CIM_ElementCapabilities — References CIM_HardwareThread and**
929 **CIM_EnabledLogicalElementCapabilities**

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** This property shall reference the instance of CIM_HardwareThread.
Capabilities	Mandatory	**Key:** This property shall reference the instance of CIM_EnabledLogicalElementCapabilities.

930 **10.5 CIM_ElementCapabilities — References CIM_Memory and**
931 **CIM_EnabledLogicalElementCapabilities**

932 CIM_ElementCapabilities associates an instance of CIM_Memory with the instance of
933 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of CIM_Memory.

934 CIM_ElementCapabilities is mandatory when the instance of CIM_Memory and the instance of
935 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of CIM_Memory
936 exist. Table 34 contains the requirements for elements of this class.

937 **Table 34 – Class: CIM_ElementCapabilities — References CIM_Memory and**
938 **CIM_EnabledLogicalElementCapabilities**

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** This property shall reference the instance of CIM_Memory.
Capabilities	Mandatory	**Key:** This property shall reference the instance of CIM_EnabledLogicalElementCapabilities.

939 **10.6 CIM_ElementCapabilities — References CIM_Processor and**
940 **CIM_ProcessorCapabilities**

941 CIM_ElementCapabilities associates an instance of CIM_Processor with the instance of
942 CIM_ProcessorCapabilities that describes the capabilities of the instance of CIM_Processor.

943 CIM_ElementCapabilities is mandatory when the instance of CIM_Processor and the instance of
944 CIM_ProcessorCapabilities exist. Table 35 contains the requirements for elements of this class.

945 **Table 35 – Class: CIM_ElementCapabilities — References CIM_Processor and**
946 **CIM_ProcessorCapabilities**

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** This property shall reference the instance of CIM_Processor.
Capabilities	Mandatory	**Key:** This property shall reference the instance of CIM_ProcessorCapabilities.

947 **10.7 CIM_ElementCapabilities — References CIM_ProcessorCore and**
948 **CIM_EnabledLogicalElementCapabilities**

949 CIM_ElementCapabilities associates an instance of CIM_ProcessorCore with the instance of
950 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of
951 CIM_ProcessorCore.

952 CIM_ElementCapabilities is mandatory when the instance of CIM_ProcessorCore and the instance of
953 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of
954 CIM_ProcessorCore exist. Table 36 contains the requirements for elements of this class.

955 **Table 36 – Class: CIM_ElementCapabilities — References CIM_ProcessorCore and**
956 **CIM_EnabledLogicalElementCapabilities**

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** This property shall reference the instance of CIM_ProcessorCore.
Capabilities	Mandatory	**Key:** This property shall reference the instance of CIM_EnabledLogicalElementCapabilities.

957 ## 10.8 CIM_EnabledLogicalElementCapabilities

958 CIM_EnabledLogicalElementCapabilities represents the capabilities of the memory, the processor core,
959 or the hardware thread. Table 37 contains the requirements for elements of this class.

960 **Table 37 – Class: CIM_EnabledLogicalElementCapabilities**

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
RequestedStatesSupported	Mandatory	See 7.6.1.1, 7.7.1.1, and 7.8.1.1.
ElementNameEditSupported	Mandatory	See 7.6.1.2, 7.7.1.2, and 7.8.1.1.
MaxElementNameLen	Conditional	See 7.6.1.3, 7.7.1.3, and 7.8.1.3.

961 ## 10.9 CIM_HardwareThread

962 CIM_HardwareThread represents the hardware thread of the processor. Table 38 contains the
963 requirements for elements of this class.

964 **Table 38 – Class: CIM_HardwareThread**

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
EnabledState	Mandatory	See 7.7.4.
RequestedState	Mandatory	See 7.7.3.
OperationalStatus	Mandatory	None
HealthState	Mandatory	None
ElementName	Mandatory	The property shall match the pattern ".*".
RequestStateChange( )	Conditional	See 8.3.

965 ## 10.10 CIM_Memory

966 CIM_Memory represents the CPU's cache memory. Table 39 contains the requirements for elements of
967 this class.

968 **Table 39 – Class: CIM_Memory**

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
CreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
DeviceID	Mandatory	**Key**
BlockSize	Mandatory	None
NumberOfBlocks	Mandatory	None
EnabledState	Mandatory	See 7.8.4.
RequestedState	Mandatory	See 7.8.3.
HealthState	Mandatory	None
OperationalStatus	Mandatory	None
ElementName	Mandatory	The property shall match the pattern ".*".
RequestStateChange( )	Conditional	See 8.4.

969 ## 10.11 CIM_Processor

970 CIM_Processor represents the processor or CPU. Table 40 contains the requirements for elements of this
971 class.

972   **Table 40 – Class: CIM_Processor**

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	**Key**
SystemName	Mandatory	**Key**
CreationClassName	Mandatory	**Key**
DeviceID	Mandatory	**Key**
Family	Mandatory	None
CurrentClockSpeed	Mandatory	When the EnabledState property has a value of 2 (Enabled), a value of 0 shall indicate that the property value is unknown. When the EnabledState property has a value of 3 (Disabled), this property shall have no meaning.
MaxClockSpeed	Mandatory	When the EnabledState property has a value of 2 (Enabled), a value of 0 shall indicate that the property value is unknown. When the EnabledState property has a value of 3 (Disabled), this property shall have no meaning.
ExternalBusClockSpeed	Mandatory	When the EnabledState property has a value of 2 (Enabled), a value of 0 shall indicate that the property value is unknown. When the EnabledState property has a value of 3 (Disabled), this property shall have no meaning.
CPUStatus	Mandatory	See 7.5.1.
EnabledState	Mandatory	See 7.5.2.
RequestedState	Mandatory	See 7.4.
OperationalStatus	Mandatory	None
HealthState	Mandatory	None
ElementName	Mandatory	The property shall match the pattern ".*".
OtherFamilyDescription	Conditional	This property shall be implemented if the Family property contains the value "Other".
RequestStateChange( )	Conditional	See 8.1.

973 ## 10.12 CIM_ProcessorCapabilities

974 CIM_ProcessorCapabilities represents the capabilities of the processor. Table 41 contains the
975 requirements for elements of this class.

976   **Table 41 – Class: CIM_ProcessorCapabilities**

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
NumberOfProcessorCores	Mandatory	A value of 0 shall mean "Unknown".
NumberOfHardwareThreads	Mandatory	A value of 0 shall mean "Unknown".

Elements	Requirement	Notes
RequestedStatesSupported	Mandatory	See 7.2.2.
ElementNameEditSupported	Mandatory	See 7.2.4.
MaxElementNameLen	Conditional	See 7.2.5.

977 **10.13 CIM_ProcessorCore**

978 CIM_ProcessorCore represents the core of the processor. Table 42 contains the requirements for
979 elements of this class.

980 **Table 42 – Class: CIM_ProcessorCore**

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
CoreEnabledState	Mandatory	See 7.6.4.1.
EnabledState	Mandatory	See 7.6.4.2.
RequestedState	Mandatory	See 7.6.3.
OperationalStatus	Mandatory	None
HealthState	Mandatory	None
ElementName	Mandatory	The property shall match the pattern ".*".
RequestStateChange( )	Conditional	See 8.2.

981 **10.14 CIM_RegisteredProfile**

982 The CIM_RegisteredProfile class is defined by the *Profile Registration Profile*. The requirements denoted
983 in Table 43 are in addition to those mandated by the *Profile Registration Profile*.

984 **Table 43 – Class: CIM_RegisteredProfile**

Elements	Requirement	Notes
RegisteredName	Mandatory	This property shall have a value of "CPU".
RegisteredVersion	Mandatory	This property shall have a value of "1.0.1".
RegisteredOrganization	Mandatory	This property shall have a value of 2 (DMTF).

985 NOTE   Previous versions of this document included the suffix "Profile" for the RegisteredName value. If
986 implementations querying for the RegisteredName value find the suffix "Profile", they should ignore the suffix, with
987 any surrounding white spaces, before any comparison is done with the value as specified in this document.

988 **10.15 CIM_SystemDevice**

989 CIM_SystemDevice associates an instance of CIM_Processor with the instance of CIM_ComputerSystem
990 of which the CIM_Processor instance is a member. Table 44 contains the requirements for elements of
991 this class.

992 **Table 44 – Class: CIM_SystemDevice**

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** This property shall reference the instance of CIM_ComputerSystem of which the instance of CIM_Processor is a member.
PartComponent	Mandatory	**Key:** This property shall reference the instance of CIM_Processor.

993

994 <div align="center">**ANNEX A**</div>
995 <div align="center">(informative)</div>
996
997 <div align="center">**Change log**</div>

Version	Date	Description
1.0.0c	2006-07-02	Preliminary Version of the Profile
1.0.0	2008-10-31	Final Version of the Profile
1.0.1	2010-04-22	Released as DMTF Standard — Changed ExternalClockSpeed to ExternalBusClockSpeed in use cases to be in sync with the MOF
1.0.2	2015-05-22	Release as DMTF Standard

998

999